sklearnapi: scikit-learn Style Module for Self-Organizing Maps

Description - The SOM Class

The `SOM` class in this module is a scikit-learn compatible estimator that wraps the self-organizing map (SOM) routines defined in the maputils module. It builds a SOM from the training data and if labels (`y`) are provided—uses them for reporting (for example, via majority voting within clusters). In the absence of labels, clusters are assigned numeric labels automatically.

Usage

```
from popsom7.sklearnapi import SOM
import pandas as pd
from sklearn import datasets

iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.DataFrame(iris.target_names[iris.target],columns=['species'])

# Initialize the SOM model with desired parameters
som = SOM(xdim=20, ydim=15, train=100000, seed=42)

# Fit the SOM model
som.fit(X, y)

# Display a summary of training parameters and quality assessments
som.summary()

# Visualize the SOM using a starburst representation
som.starburst()
```

Class Parameters

- **xdim** : *int, default=10*
  Number of neurons along the x-dimension of the map.
- **ydim** : *int, default=5*
  Number of neurons along the y-dimension of the map.
- **alpha** : *float, default=0.3*
  Learning rate, a value between 0 and 1.
- **train** : *int, default=1000*
  Number of training iterations.
- **normalize** : *bool, default=False*
  Indicates whether the input data should be normalized by row.

- **seed** : *int or None, default=None*
  Seed value for reproducibility.

Attributes

- **som_map_** : *dict*
  The fitted SOM model. For details on the model check the documentation of the function map_build.

Methods

___init___(xdim=10, ydim=5, alpha=0.3, train=1000, normalize=False, seed=None)

Description:
Initializes the SOM instance with the specified parameters.

Usage:
Called automatically when creating a new `SOM` object.

---

fit(X, y=None)

Description:
Fits the SOM model using the provided training data.

Usage:

```
som.fit(X, y)
```

Arguments:

- **X** : array-like or `pd.DataFrame`, shape *(n_samples, n_features)*
  Training data.
- **y** : array-like or `pd.DataFrame`, shape *(n_samples,)* or *(n_samples, 1)*, default=None
  Optional labels for the training data (used for reporting).

Returns:

- `self` : SOM instance
  The fitted model.

---

fit_predict(X, y=None)

Description:
Fits the SOM model and returns cluster assignments for the training data.

Usage:

```
labels = som.fit_predict(X, y)
```

Arguments:

- **X** : array-like or `pd.DataFrame`, shape *(n_samples, n_features)*
  Training data.
- **y** : array-like or `pd.DataFrame`, default=None
  Optional labels for the training data.

Returns:

- `labels` : array-like
  Cluster labels for each training sample.

---

predict(X)

Description:
Maps new samples to clusters using the fitted SOM model.

Usage:

```
predicted_labels = som.predict(X)
```

Arguments:

- **X** : array-like or `pd.DataFrame`, shape *(n_samples, n_features)*
  New data to be classified.

Returns:

- `labels` : `pd.Series` or list
  Cluster labels assigned to each sample.

Notes:
Raises a `ValueError` if the model has not been fitted.

---

transform(X)

Description:
Maps input samples to their corresponding (x, y) positions on the SOM grid.

Usage:

```
positions = som.transform(X)
```

Arguments:

- **X** : array-like or `pd.DataFrame`, shape *(n_samples, n_features)*
  New data.

Returns:

- `positions` : `pd.DataFrame`
  A DataFrame with columns `"x-dim"` and `"y-dim"` indicating the grid positions.

---

summary(verbose=True)

Description:
Prints and returns a summary of the SOM training parameters and quality assessments.

Usage:

```
summary_info = som.summary(verbose=True)
```

Arguments:

- **verbose** : *bool, default=True*
  If `True`, prints the summary to the screen.

Returns:

- `summary_info` : *dict*
  A dictionary containing summary information about the SOM model.

---

starburst()

Description:
Displays a starburst (heat map) representation of the SOM, visualizing clusters on the map.

Usage:

```
som.starburst()
```

---

significance(graphics=True, feature_labels=True)

Description:
Computes the relative significance of each feature and optionally generates a plot.

Usage:

```
sig_values = som.significance(graphics=True, feature_labels=True)
```

Arguments:

- **graphics** : *bool, default=True*
  If `True`, produces a plot.

- **feature_labels** : *bool, default=True*
  If `True`, uses feature names in the plot.

Returns:

- `significance` : array-like or None
  A vector of feature significance values (if `graphics` is `False`).

---

marginal(marginal)

Description:
Displays a density plot for a chosen dimension overlaid with neuron density.

Usage:

`som.marginal(marginal)`

Arguments:

- **marginal** : *int or str*
  The index or name of the data frame column to be visualized.

Example

```
import pandas as pd
from som_module import SOM

# Load dataset (e.g., the Iris dataset)
iris = pd.read_csv('iris.csv')
X = iris.drop(columns=['id', 'Species'])
y = iris[['Species']]  # Optional: for reporting purposes

# Create and fit the SOM model
som = SOM(xdim=20, ydim=15, alpha=0.3, train=100000, seed=42)
som.fit(X, y)

# Display a summary of the SOM model
som.summary()

# Get cluster assignments for the training data
labels = som.fit_predict(X, y)

# Predict clusters for new data
predicted_labels = som.predict(X)

# Obtain the grid positions of data samples
```

```
positions = som.transform(X)

# Visualize the SOM with a starburst plot
som.starburst()

# Plot feature significance
sig_values = som.significance()

# Generate a marginal plot for a specific feature
som.marginal("Petal.Length")
```